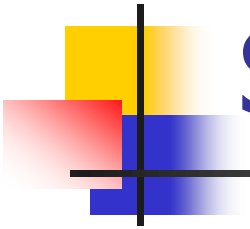
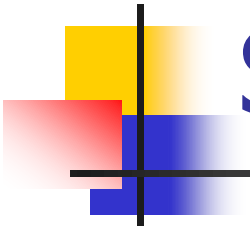


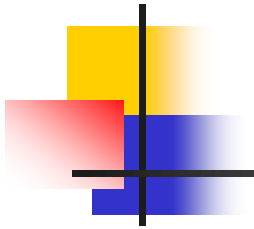
Système d'exploitation

- Introduction
- Systèmes :
  - Concepts des Systèmes
  - Types des Systèmes
  - Noyau du Système
  - Gestion Processus
  - Gestion Fichiers



Système d’exploitation

- Commandes Linux(ubuntu)
- Programmation Système (langage C)
- Programmation des Scripts(Bash)
- Projets



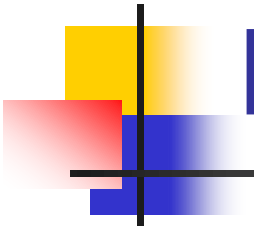
- Serveur
- Système Raid
- Carte Raid
- Partie logicielle

Serveur

- Matériel :CPU :Xéon, capacité mémoire élevée, Plusieurs DD SCSI(#IDE), Système serveur, +cartes réseaux
- Logiciel : serveur DNS traduction des noms de domaines en adresses ip et inversement
- Serveur du domaine :

Serveur principal

- Création des comptes
- Création des groupes : utilisateurs, machines
- Active directory(windows server)
- Profile utilisateur
- LDAP (linux)

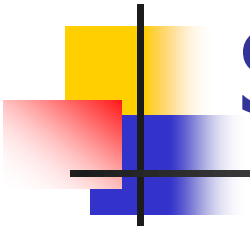


Domaine

- Plusieurs domaines
- approbation entre domaine
- Partage des ressources

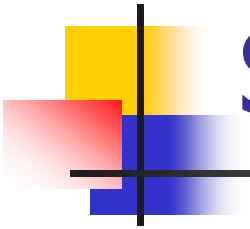
DNS

- http://adresse_ip
- http://www.fsts.ac.ma
- /etc/hosts
- adresses_ip Nom_machine



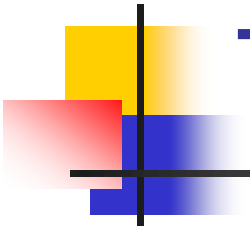
Serveur DHCP

- Distribution des adresses ip
- ddns
- Serveur doit avoir adresse statique



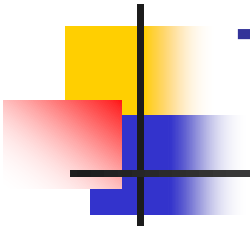
Serveur Web

- Apache
- Webshère



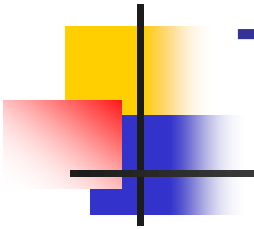
Types systèmes

- Monotache
- Multitaches
- Multiutilisateurs



Type système

- Sysstème Batch
- Système Temps partagé
- Sysstème Temps réel: OS9, RTLINUX,QNX
- Le temps réel est la capacité à répondre à une sollicitation en un temps déterminé pour produire une réaction appropriée

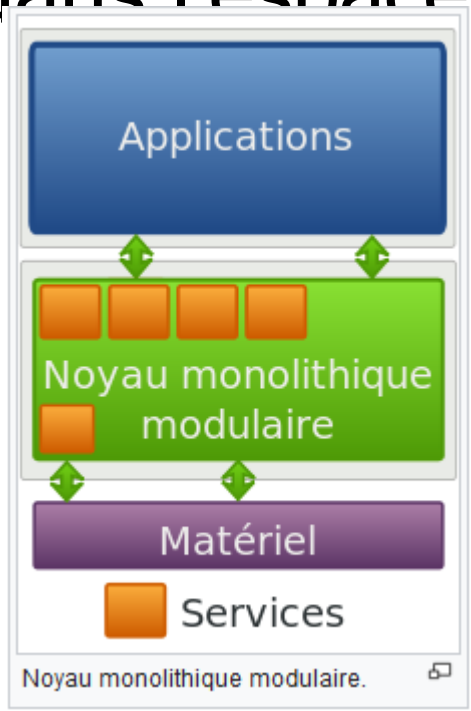


Type système

- Système embarqué
- Système monoposte
- Système réseau

Noyau monolithique

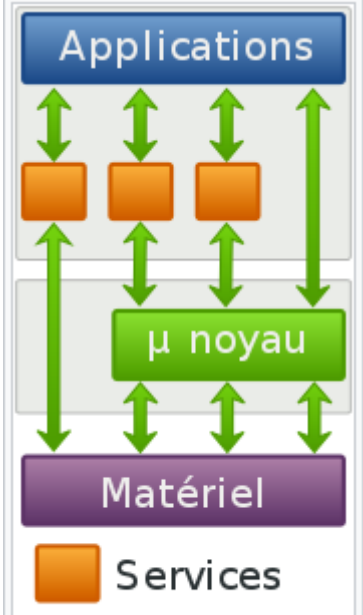
- placent un maximum de programmes systèmes dans l'espace noyau.



Noyau monolithique modulaire.

micro-noyaux

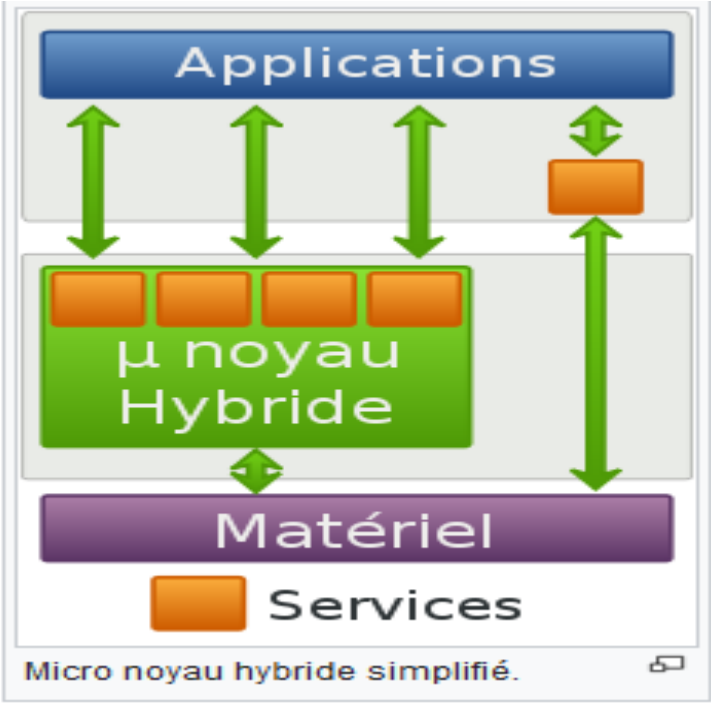
- préfèrent au contraire placer le plus de choses dans l'espace utilisateur.



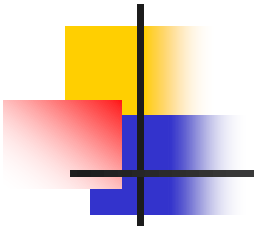
Micro noyau simplifié.

noyaux hybrides

- sont un intermédiaire entre les deux précédents

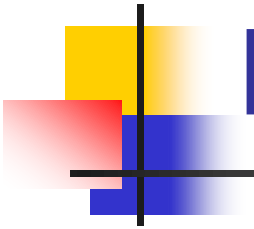


Micro noyau hybride simplifié.



Définition

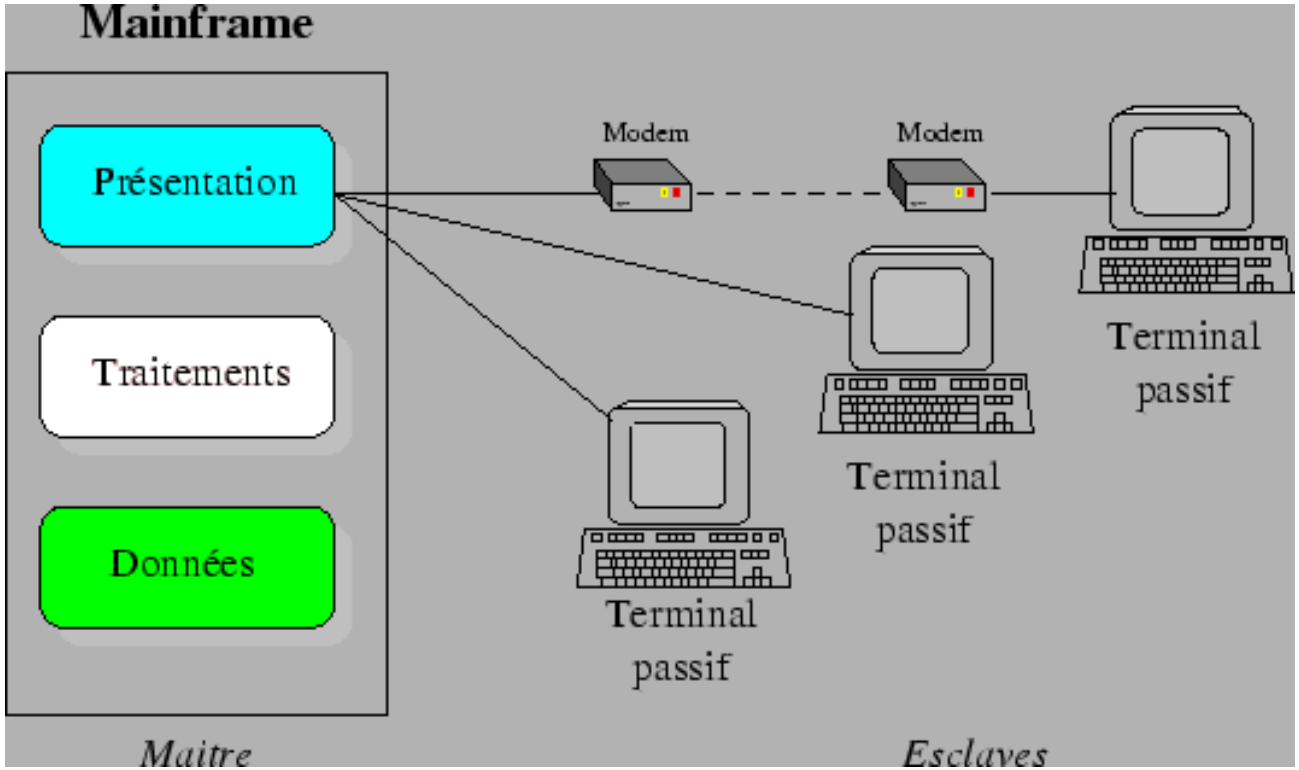
Ensemble de programmes permettant la gestion optimale des ressources de la machine et fournissant aux programmes utilisateurs une interface simplifiée.
Ressources : matérielles et logicielles



Définitions

- Multiprogrammation :possibilité de charger plusieurs programme en mémoire
- Algorithmes allocation: First Fit, Best Fit, Worst Fit
- Systèmes distribués :ensemble de machines connectées, chaque machine exécute des processus et coordonne ses activités avec les autres machines
- Systèmes parallèles : système qui prennent en charge plusieurs processeurs.
- Cray I , II

Le serveur central prend en charge l'intégralité des traitements, y compris l'affichage qui est simplement déporté sur des terminaux passifs.



Rôles d'un système d'exploitation

Problème 1 : Satisfaire un ensemble d'utilisateurs ayant des besoins importants

- de traitement :
- d' archivage et de communications telnet, ssh,…
- telnet adresse_ip
- Communication entre processus: Pipe, Socket, …

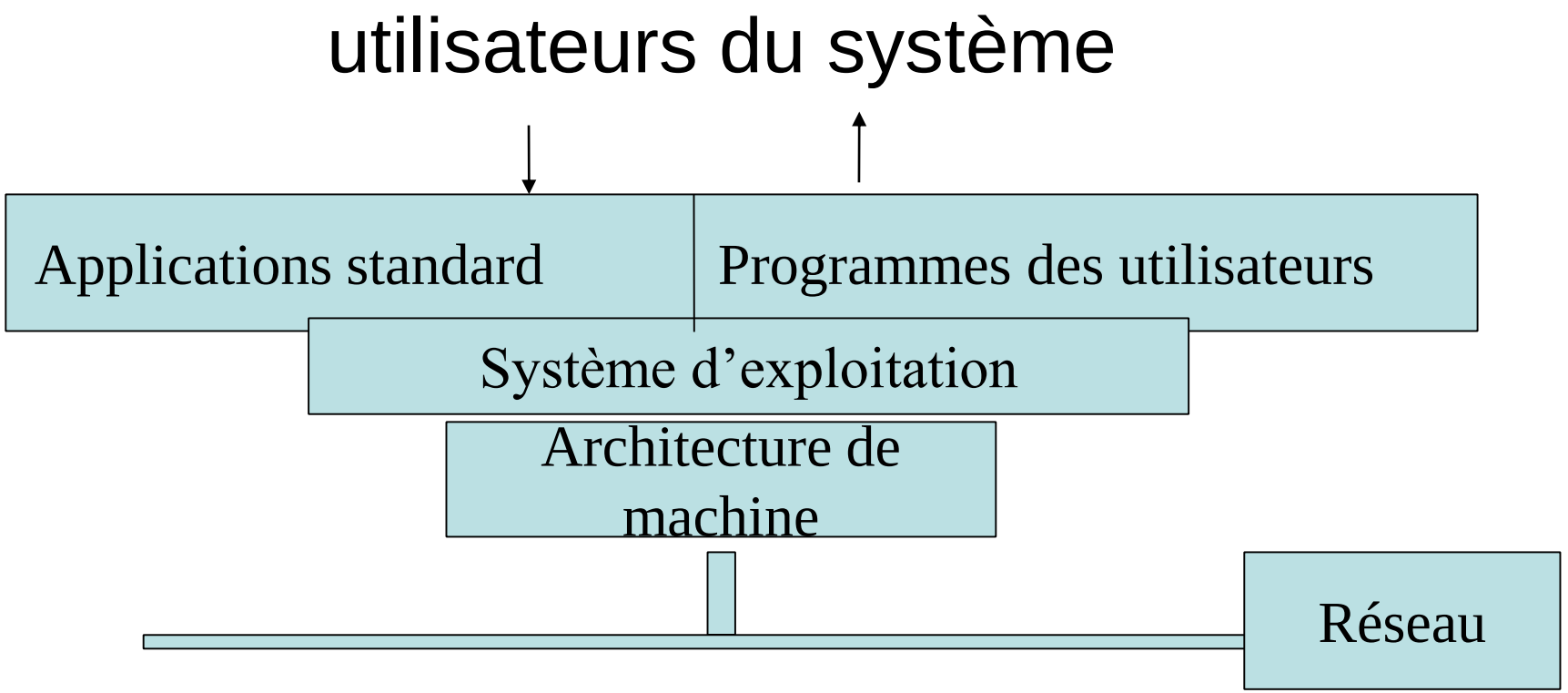
Problème 2 : Contrôler un ensemble complexe et évolutif :

- De machines et de mémoires
- De périphériques et de connections
- De logiciels

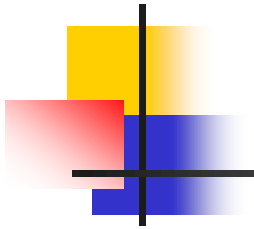
Problèmes 3 : Réaliser un compromis convenable entre :

- La qualité des services offerts aux utilisateurs
- L'optimisation des ressources du centre de traitement
- Le système est l'intermédiaire obligatoire et indispensable entre les utilisateurs et la configuration de machines.

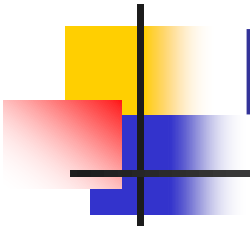
Structure d'un système d'exploitation



Structure d'un système d'exploitation (suite)

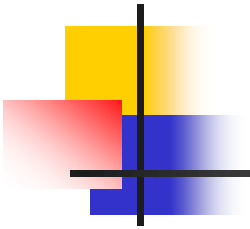


- Services haut niveau : tout programme que l'utilisateur peut utiliser directement
- Superviseur : sous partie du noyau
- Edition de lien
- Noyau : programme ayant accès aux matériels
- Programmes noyau: démons



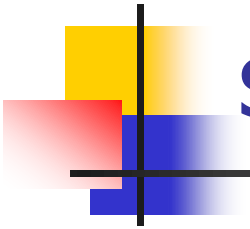
Mode système d'exploitation

- Mode user : tâches utilisateurs
- Mode Kernel: sécurité système, création processus,…

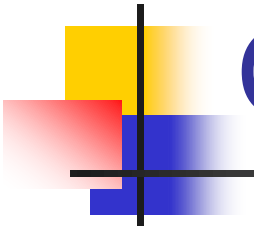


- Compilation :
- Prog source----→prog objet
- A+b en ASCII
- + adresse a adresse b
- Edition de lien:
- prog objet---→exécutable

Fonctions principales d'un système d'exploitation



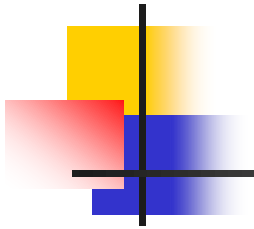
- Gestion des processus
- Gestion de la mémoire
- Gestion des fichiers
- Gestion des entrées sorties: DMA processeur d'E/S
- Int N



Gestion des processus

- Création Processus
- Ordonnancement des processus
- Communication interprocessus

Processus et Threads



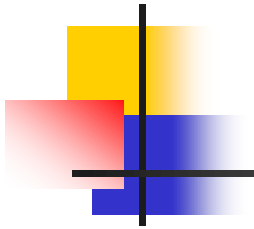
- Déf 1: Un "Processus " est un programme en cours d'exécution.
- Déf 2: Un "Processus" est l'exécution séquentielle des instructions d'un programme dans un espace de travail avec un ensemble de valeurs de variables.

Un processus a un identificateur (numéro) PID

'Unique ' qui est donnée par le système quand il crée le processus

Déf 3: Un Thread (fil d'exécution) est un processus léger partageant le même segment d'adresse.

Creation de processus



- Evénements provoquant la création d'un processus
- - Initialisation du système
- - Exécution d'un appel système de création de processus en cours d'exécution
- =>Les processus peuvent créer d ’autres processus, formant une hiérarchie (instruction fork() ou semblables)
- - Requête utilisateur solicitant la création d'un nouveau processus system(), execl()

Création des processus

- Hiérarchie des processus

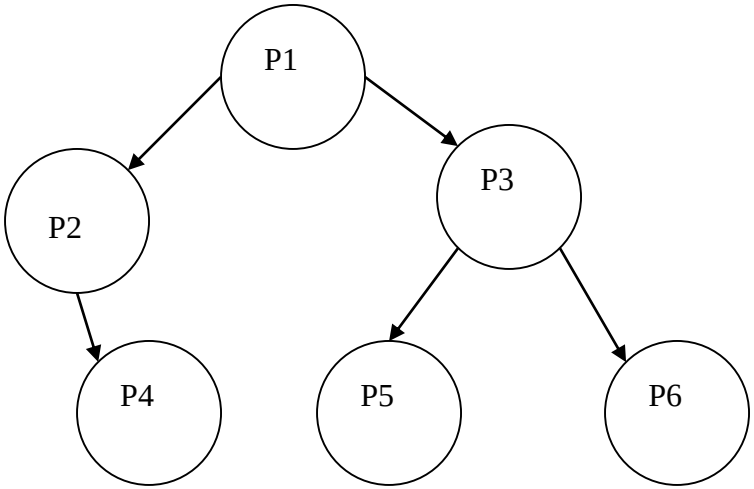
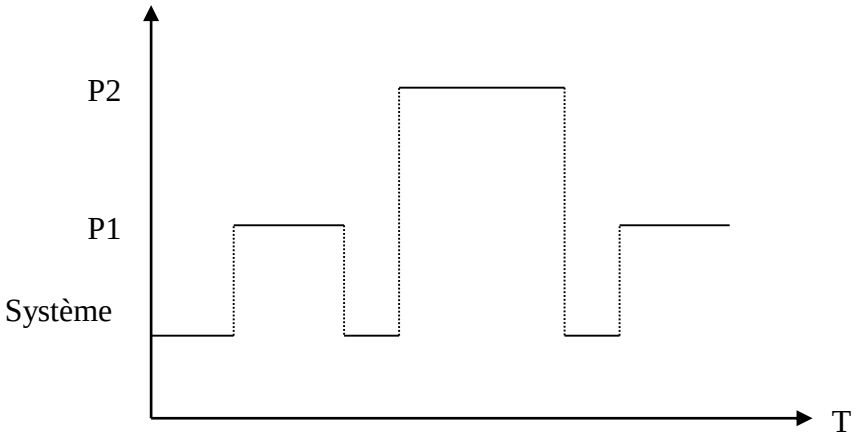


Fig. 1. Hiérarchie des processus

Parallélisme



DÉFINITION

- Plusieurs définitions

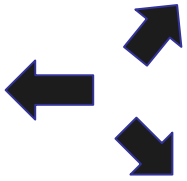
1- Programme en exécution

Programme lancé mais non termin

- Encore présent dans le système
- Peut être dans un état quelconque

2- Entité dynamique

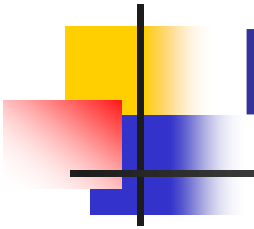
- Programme La Vieille Dame Et Ses Gâteaux
  - Entité statique
  - Texte interprétable
- Processus
  - Entité dynamique
  - Abstraction de l'exécution d'un programme



3- Suite d'états

E0 → E1 → … → Ek→ …

- Etat initial puis convergence vers un état final



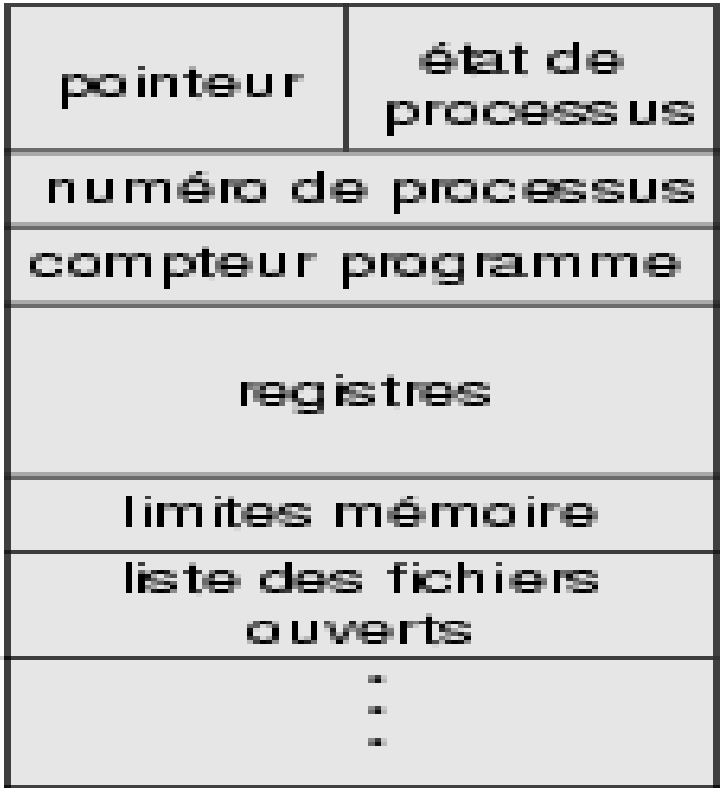
Processus

L'espace de travail comporte :

- ***Le code exécutable du programme***
- ***La pile d'exécution***
- ***Les variables et les constantes partagées***
- ***Ressources: Les fichiers ouverts***
- ***Chaque processus est représenté par un PCB***

PCB = Process Control Block:

Représente la situation actuelle d 'un processus, pour le reprendre plus tard



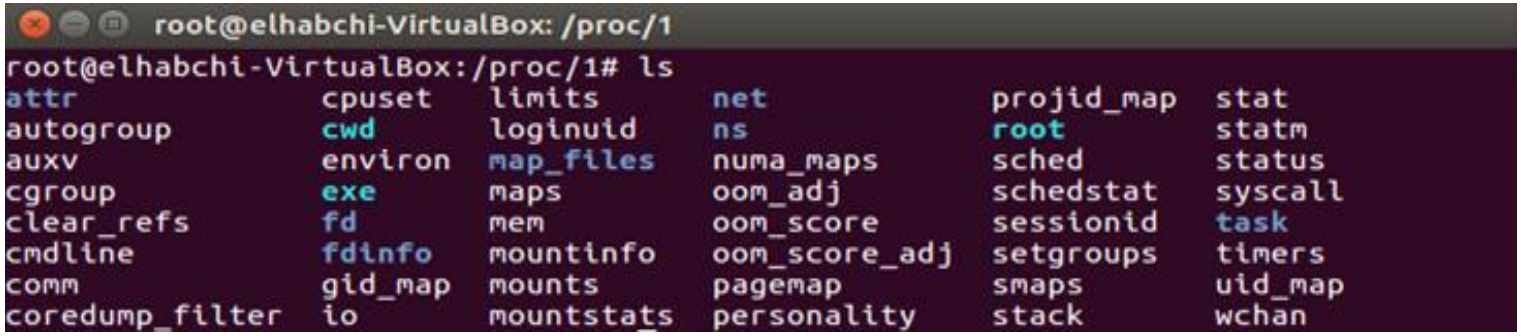
TYPES DE PROCESSUS

Types de processus

- Processus en avant plan
  - Foreground (visible)
  - Processus avec lequel on dialogue
    - Dialogue à travers le clavier et l'écran
- Processus en arrière plan
  - Background
  - S'exécute sans dialogue avec l'utilisateur
    - Tâche de fond (Job)

STRUCTURE DE DONNÉES

- Le pseudo système de fichiers /proc est monté au démarrage du système d'exploitation. Il contient entre autres, les informations précédentes.

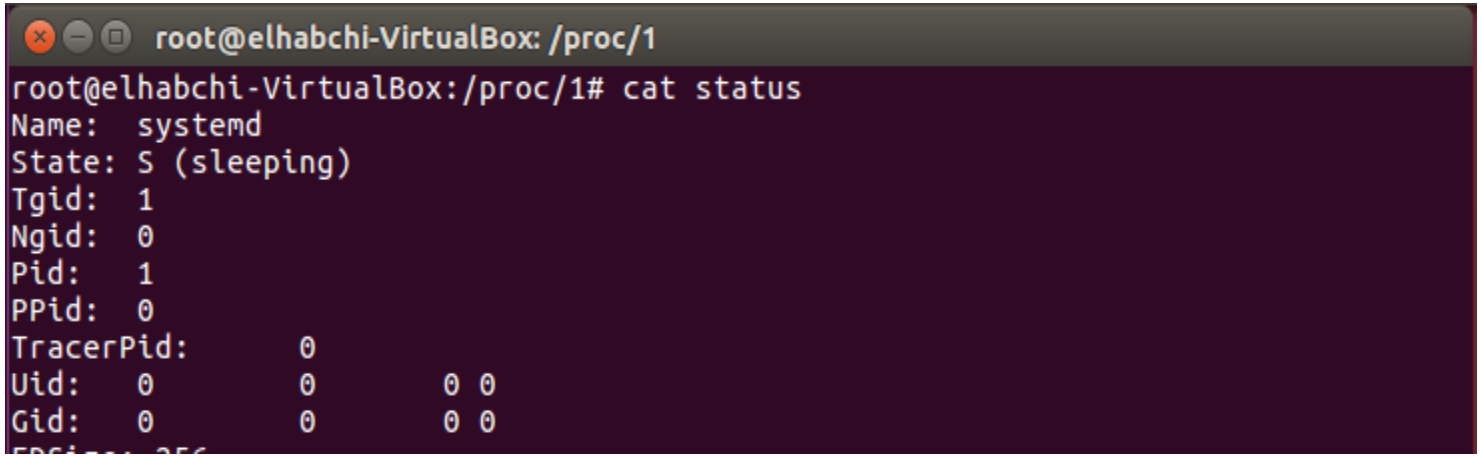


→ **Les informations vitales à l'exécution du processus sont stockées dans ces fichiers.**

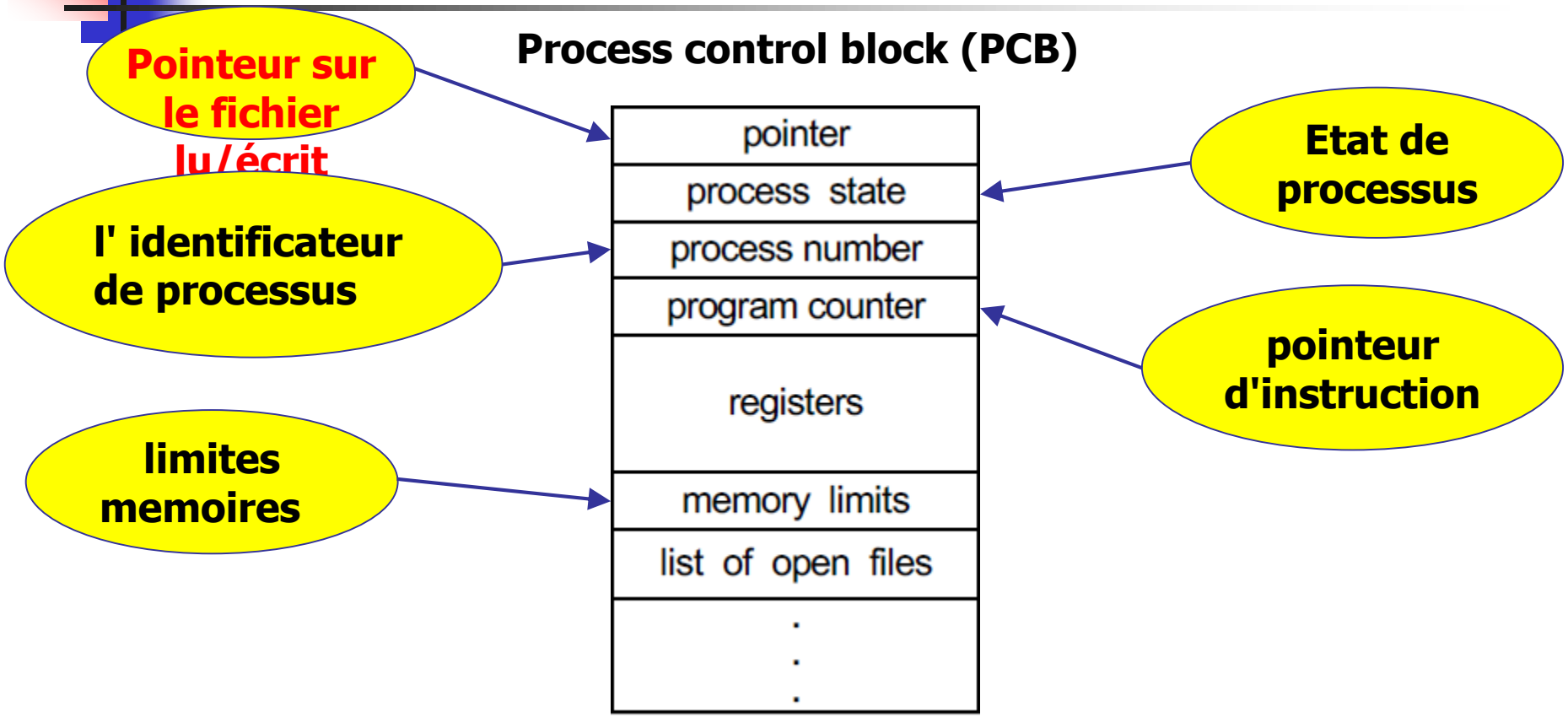
STRUCTURE DE DONNÉES

Par exemple:

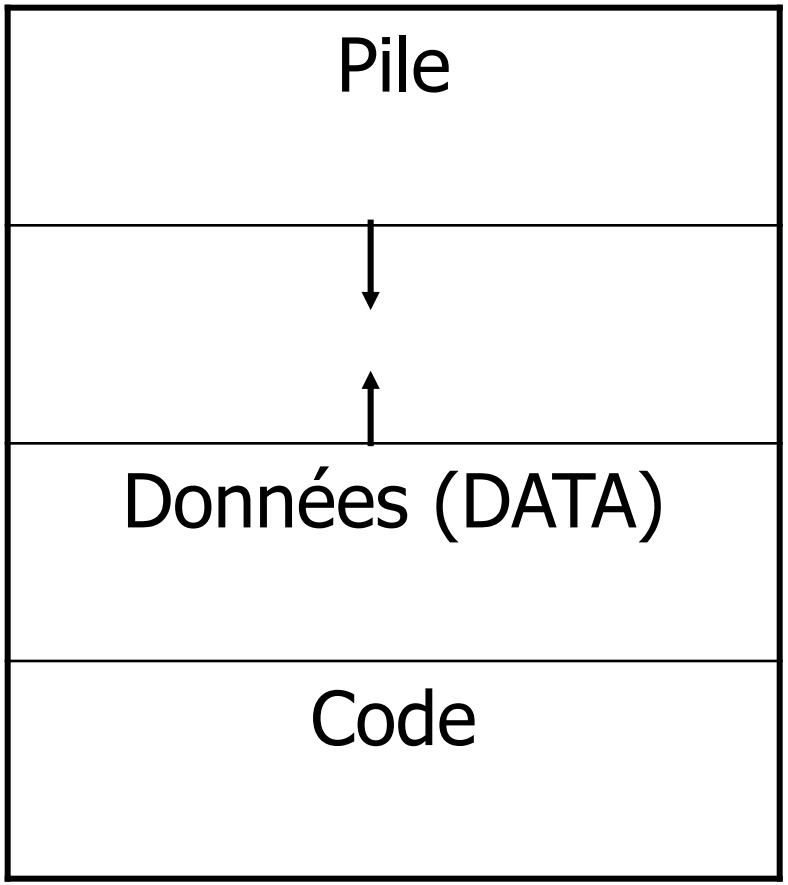
status : c'est un fichier qui donne des informations sur le statut actuel du processus



DÉFINITION


Process control block (PCB)
Pointeur sur le fichier lu/écrit
l' identificateur de processus
limites memoires
Etat de processus
pointeur d'instruction
pointer
process state
process number
program counter
registers
memory limits
list of open files

Segments d'un processus



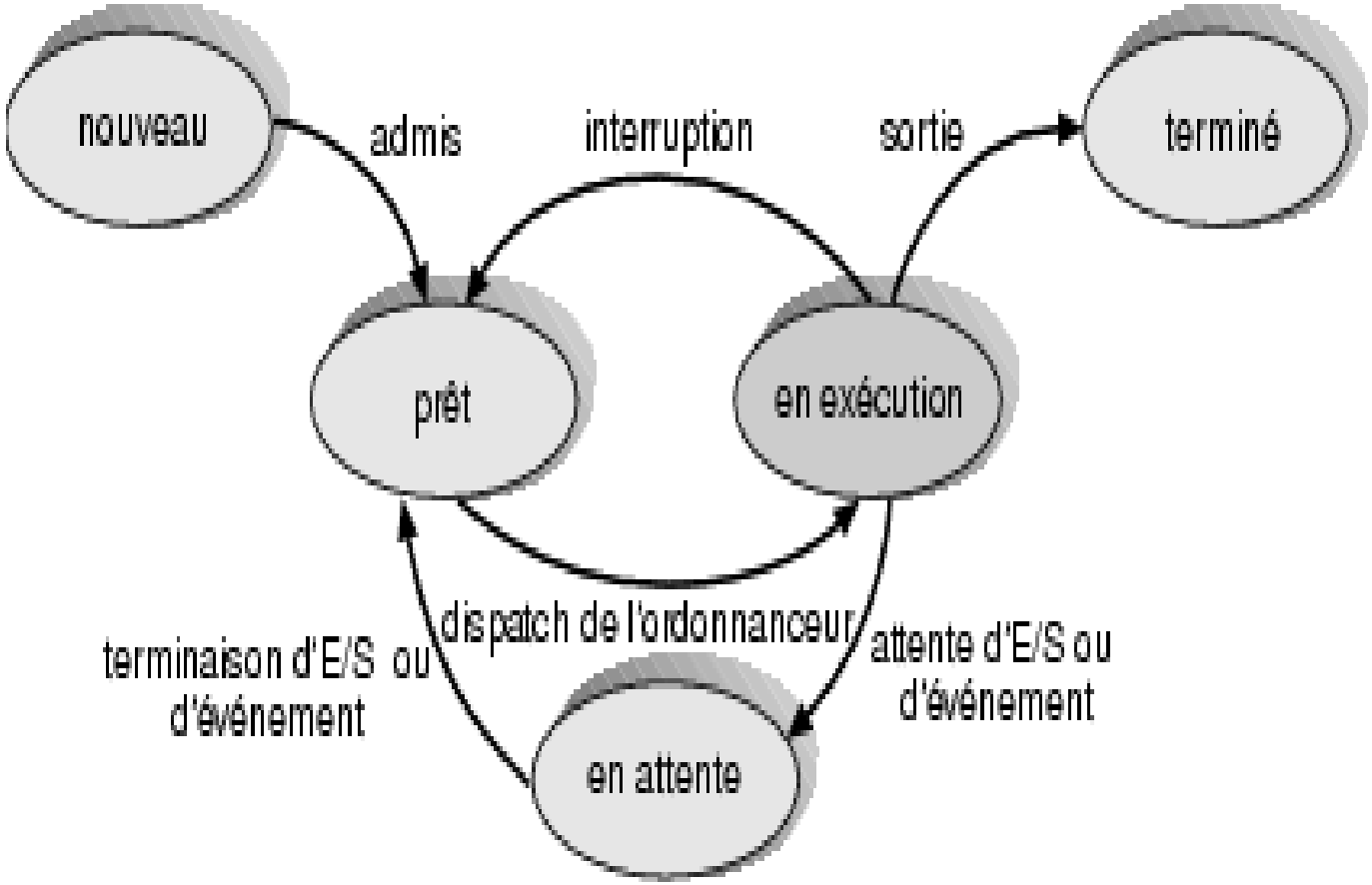
Terminaison de processus

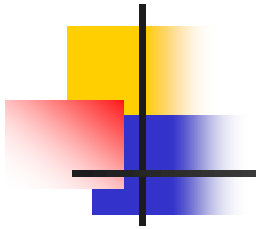
- Un processus exécute sa dernière instruction
  - pourrait passer des données à son parent
  - ses ressources lui sont enlevées
- Le parent termine l'exécution d'un fils pour raisons différentes
  - le fils a excédé ses ressources
  - le fils n`est plus requis
- etc.

État de processus

- Au fur et a mesure qu'un processus s'exécute, il change d'état
  - nouveau: le processus vient d 'être créé
  - exécutant-running: le processus est en train d 'être exécuté par l 'UCT
  - attente-waiting: le processus est en train d 'attendre un événement (p.ex. la fin d 'une opération d 'E/S)
  - prêt-ready: le processus est en attente d'être exécuté par l 'UCT
  - terminated: fin d 'exécution

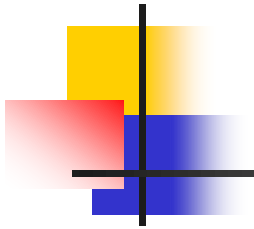
Diagramme de transition d`états d`un processus





- **<u>Contexte processus et contexte système :</u>**

  Le kernel peut opérer en contexte processus autrement dit pour le compte d'un processus (par appel système). Il peut aussi opérer en contexte système pour gérer les interruptions en provenance des périphériques.



ORDONNANCEMENT

ORDONNANCEMENT

- Chaque fois, que le processeur devient inactif, le système d'exploitation doit sélectionner un processus de la file d'attente des processus prêts, et lui passe le contrôle.

➔ On distingue deux cas :

- le Dispatcheur
- le Scheduleur (Ordonnanceur)

ORDONNANCEMENT

❖ Le Dispatcheur

Il s'occupe de l'allocation du processeur à un processus sélectionné par l'Ordonnanceur du processeur. Une fois allouer, le processeur doit réaliser les tâches suivantes :

Commutation de contexte :sauvegarder le contexte du processus qui doit relâcher le processeur et charger le contexte de celui qui aura le prochain cycle processeur

Commutation du mode d'exécution :basculer du mode Maître (mode d'exécution du dispatcheur) en mode utilisateur (mode d'exécution du processeur utilisateur)

Branchement :se brancher au bon emplacement dans le processus utilisateur pour le faire démarrer.

ORDONNANCEMENT

❖ L'Ordonnanceur

Certains systèmes d'exploitation utilisent une technique d'ordonnancement à deux niveaux qui intègre deux types d'Ordonnanceurs :

Ordonnanceur du processeur : c'est un Ordonnanceur court terme opère sur un ensemble du processus présents en mémoire. Il s'occupe de la sélection du processus qui aura le prochain cycle processeur,à partir de la file d'attente des processus prêts.

Ordonnanceur de travail : ou Ordonnanceur long terme, utilisé en cas d'insuffisance de mémoire, son rôle est de sélectionner le sous ensemble de processus stockés sur un disque et qui vont être chargés en mémoire. Ensuite, il retire périodiquement de la mémoire les processus qui sont restés assez longtemps et les remplace par des processus qui sont sur le disque depuis trop de temps.

ORDONNANCEMENT

❖ Algorithme d'ordonnancement

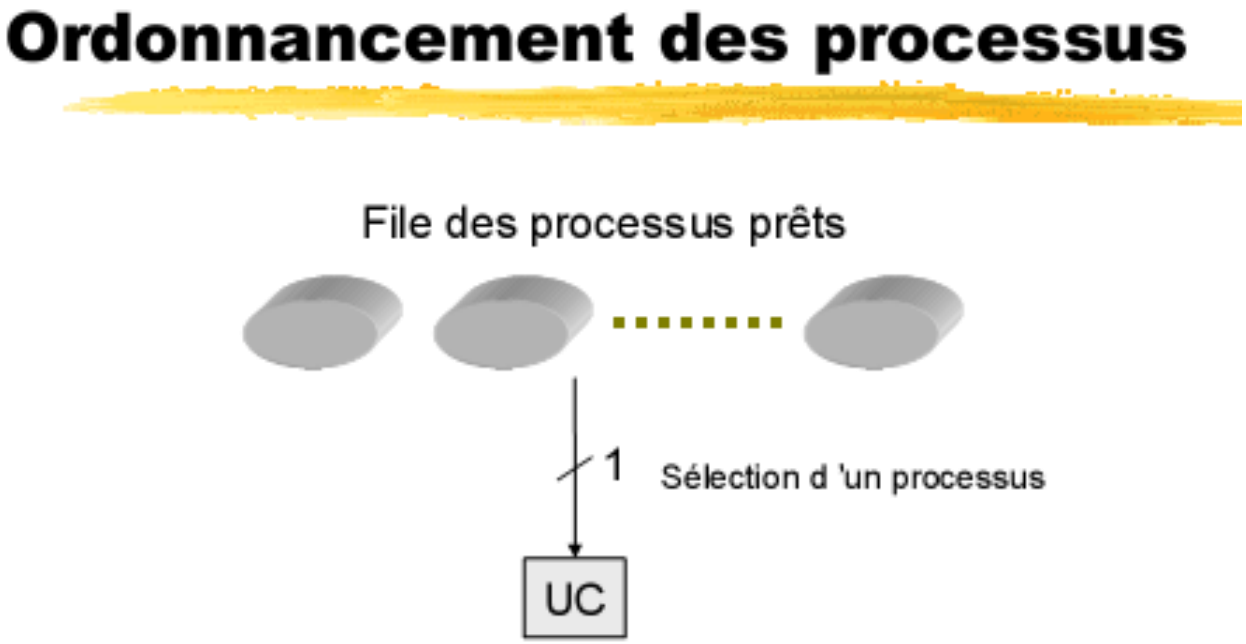
Linux est un système à temps partagé. Bien que le processeur ne puisse exécuter qu'une tâche à la fois, il faut que l'utilisateur ait l'impression que tous les processus sont exécutés en même temps. Le scheduler utilise trois techniques pour réaliser cela :

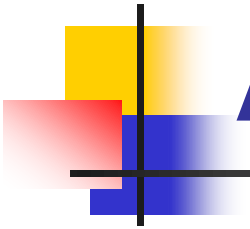
➢ **quantum**: un intervalle de temps au bout duquel le processus en exécution s'arrête et le scheduler regarde à qui il va maintenant donner le processeur.

➢ **préemptive** : le scheduler peut interrompre l'exécution d'un processus non-noyau pour donner le processeur à un autre processus, et ce même au milieu d'un quantum.

➢ **priorité** : chaque processus possède une priorité comprise entre -20 et +19. Plus la priorité est basse plus le scheduler privilégiera ce processus.

Ordonnancement de processus :





Algorithmes d'ordonnancement

Nous distinguons plusieurs algorithmes d'ordonnancement, les plus répandus sont :

- Ordonnancement Premier Arrivé Premier Servi (FCFS)
- Ordonnancement du plus court d'abord (SJF)
- Ordonnancement circulaire : Tourniquet (RR)
- Ordonnancement avec priorité

L'ordonnancement est la partie du système d'exploitation qui détermine dans quel ordre les processus prêts à s'exécuter (présents dans la file des prêts) seront élus.

Ses objectifs sont :

- Assurer le plein usage du CPU (agir en sorte qu'il soit le moins souvent possible inactifs);
- Réduire le temps d'attente des utilisateurs.
- Assurer l'équité entre les utilisateurs.

Un algorithme d'ordonnancement permet d'optimiser une des grandeurs temporelles suivantes :

- Le temps de réponse moyen décrit la moyenne des dates de fin d'exécution :

$$TRM = \sum_{i=1}^{n} TRi \Big/ n$$, avec TR_i=date fin –date arrivée.

- Le temps d'attente moyen est la moyenne des délais d'attente pour commencer une exécution :

$$TAM = \sum_{i=1}^{n} TAi \Big/ n$$, avec $TA_i = TR_i$ - temps d'exécution

Les algorithmes d'ordonnancement se distinguent les uns des autres du fait que certains autorisent la réquisition de l'unité centrale alors que d'autres l'interdisent. La réquisition est la possibilité de retirer à n'importe quel instant le processeur à un processus même si ce dernier est en cours d'exécution.

Ordonnancement FCFS

Dans cet algorithme ; connu sous le nom FIFO (First In, First Out) ; les processus sont rangés dans la file d'attente des processus prêts selon leur ordre d'arriver.

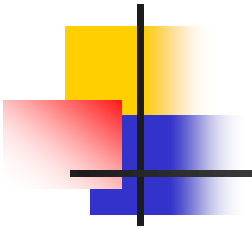
Les règles régissant cet ordonnancement sont :

1. Quand un processus est prêt à s'exécuter, il est mis en queue de la file d'attente des processus prêts.
2. Quand le processeur devient libre, il est alloué au processus se trouvant en tête de file d'attente des processus prêts.
3. Le processus élu relâche le processeur s'il se termine ou s'il demande une entrée sortie

Caractéristiques de l'Ordonnanceur

- Ordonnanceur sans réquisition
- Ordonnanceur non adapté à un système temps partagé car dans un système pareil, chaque utilisateur obtienne le processeur à des intervalles réguliers.

processus	Durée Estimé	Date Arrivée
P1	8	0
P2	4	1
P3	5	2
P4	9	3



- 0---→8---→12----→17--→26
- P1 P2 P3 P4
- P1 TR1=8-0=8 TA1=8-8=0
- P2 TR2=12-1=11 TA2=11-4=7
- P3 TR3=17-2=15 TA3=15-5=10
- P4 TR4=26-3=23 TA4=23-9=14

Ordonnancement SJF (Shortest Job First) (Plus Cours temps d'Exécution en Premier) SJF

- choisit de façon prioritaire les processus ayant le plus court temps d'exécution sans réellement tenir compte de leur date d'arrivée. Dans cet algorithme les différents processus sont rangés dans la file d'attente des processus prêts selon un ordre croissant de leur temps d'exécution (cycle de calcul). Les règles régissant cet ordonnancement sont :
- 1. Quand un processus est prêt à s'exécuter, il est inséré dans la file d'attente des processus prêts à sa position approprie.
- 2. Quand le processeur devient libre, il est assigné au processus se trouvant en tête de la file d'attente des processus prêts (ce processus possède le plus petit cycle processeur.). Si deux processus ont la même longueur de cycle, on applique dans ce cas l'algorithme FCFS.
- 3. Si le système ne met pas en oeuvre la réquisition, le processus élu relâche le processeur s'il se termine ou s'il demande une entrée sortie. Dans le cas contraire, le processus élu perd le processeur également. Quand un processus ayant un cycle d'exécution inférieur au temps processeur restant du processus élu, vient d'entrer dans la file d'attente des prêts. Le processus élu dans ce cas sera mis dans la file d'attente des éligibles, et le processeur est alloué au processus qui vient d'entrer.

Caractéristiques de l'Ordonnanceur

- Cet Ordonnanceur peut être avec ou sans réquisition
- Implémentation difficile, car il n'existe aucune manière pour connaître le cycle suivant du processeur.

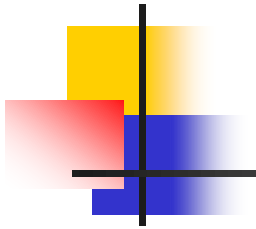
processus	Durée Estimé	Date Arrivée
P1	10	0
P2	05	2
P3	15	3
P4	03	4

Ordonnancement RR (round robin)

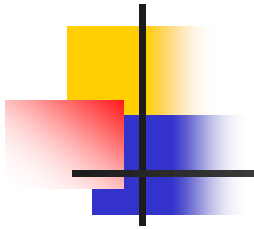
Dans cet algorithme les processus sont rangés dans une file d'attente des éligibles, le processeur est alloué successivement aux différents processus pour une tranche de temps fixe Q appelé Quantum. Cet Ordonnancement est régit par les règles suivantes :

1. Un processus qui rentre dans l'état éligible est mis en queue de la file d'attente des prêts.
2. Si un processus élu se termine ou se bloque avant de consommer son quantum de temps, le processeur est immédiatement alloué au prochain processus se trouvant en tête de la file d'attente des prêts.
3. Si le processus élu continue de s'exécuter au bout de son quantum, dans ce cas le processus sera interrompu et mis en queue de la file d'attente des prêts et le processeur est réquisitionné pour être réalloué au prochain processus en tête de cette même file d'attente

Caractéristiques de l’Ordonnanceur



- Avec réquisition
- Adapté aux systèmes temps partagé.
- La stratégie du tourniquet garantit que tous processus est servis au bout d’un temps fini. Son avantage est d’éviter la famine. On dit qu'un processus est en famine lorsqu'il est prêt à être exécuté et se voit refuser l'accès à une ressource (ici le processeur) pendant un temps indéterminé
- L’efficacité de cet ordonnanceur dépend principalement de la valeur du quantum Q: o Le choix d'un Q assez petit augmente le nombre de commutation. o Le choix d'un Q assez grand augmente le temps de réponse du système



	Quantum = 1	Quantum = 10
TRM		
TAM		

processus	Durée Estimé	Date Arrivée
P1	30	0
P2	05	1
P3	02	2

Ordonnacement SRTF (Shortest Remaining Time first)

choisit le processus dont le temps d'exécution restant est le plus court. C'est une variante de l'algorithme SJF

Cet algorithme est non implantable parce qu'il suppose, entre autres, connu le temps d'exécution réel de chaque processus pour pouvoir calculer le temps restant

processus	Durée Estimé	Date Arrivée
P1	8	0
P2	5	2
P3	5	3
P4	2	4

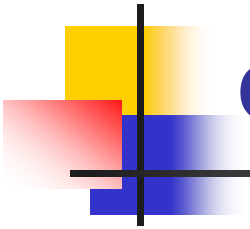
Ordonnancement basé sur les priorités

Dans cet algorithme, les processus sont rangés dans la file d'attente des processus prêt par ordre décroissant de priorité. L'ordonnancement dans ce cas est régit par les règles suivantes :

1. Quand un processus est admis par le système, il est inséré dans la file d'attente des processus prêts à sa position approprie (dépend de la valeur de priorité).

2. Quand le processeur devient libre, il est alloué au processus se trouvant en tête de file d'attente des processus prêts (le plus prioritaire).

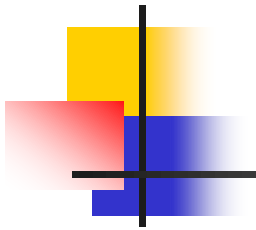
3. Un processus élu relâche le processeur s'il se termine ou se bloque (E/S ou autre).



Caractéristiques de l'Ordonnanceur

Les principales caractéristiques sont :

- Peut être avec ou sans réquisition
- Un processus de priorité basse risque de ne pas être servi (problème de famine) d'où la nécessité d'ajuster périodiquement les priorités des processus prêts. L'ajustement consiste à incrémenter graduellement la priorité des processus de la file d'attente.



processus	Durée Estimé	Priorité
P1	5	1
P2	8	5
P3	4	3

Exercices

- Soit un système à n UCT, quel est le nombre maximal de processus dans les états prêt, exécution et bloqué ?
- Soit un système avec n processus combien existe-t-il de manières d'ordonnancer ces processus ?
- Donner le schéma d'exécution pour :
- FCFS,
- SJF
- SRT,Tourniquet q=2

A	0.00	3
B	1.001	6
C	4.001	4
D	6.001	2

APPEL SYSTÈME

❖ Définition

Un appel système est implanté au sein du noyau Linux. Lorsqu'un programme effectue un appel système, les arguments sont mis en forme et transférés au noyau qui prend la main jusqu'à la fin de l'appel.

APPEL SYSTÈME

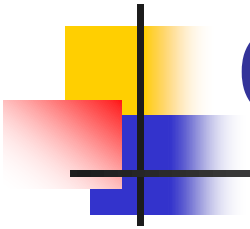
Appel système	Description
`nice( )`	change la priorité[1] d'un processus
`getpriority( )`	renvoi la priorité max d'un groupe de processus
`setpriority( )`	définit la priorité d'un groupe de processus
`sched_getscheduler( )`	renvoi la politique d'ordonnancement[2] d'un processus
`sched_setscheduler( )`	définit la politique d'ordonnancement et la priorité d'un processus
`sched_getparam( )`	renvoi la priorité d'un processus
`sched_setparam( )`	définit la priorité d'un processus
`sched_yield( )`	abandon volontaire du processeur sans blocage
`sched_get_priority_min( )`	renvoi la priorité min. pour une politique
`sched_get_priority_max( )`	renvoi la priorité max. pour une politique
`sched_rr_get_interval( )`	renvoi le quantum de temps pour une politique *Round Robin (tourniquet)*

APPEL SYSTÈME

❖ Localisation

Linux propose près de 300~appels système. La liste des appels disponibles sur votre système se trouve dans le fichier /usr/include/asm/unistd.h. Certains ne sont destinés qu'à être utilisés en interne par le noyau et d'autres ne servent que pour l'implémentation de certaines bibliothèques.

➔ La plupart sont déclarés dans le fichier d'en-tête /usr/include/asm/unistd.h



Création processus

- Int fork()
- Retourne : 0 pour le nouveau processus
- Pid du nouveau pour l'ancien processus
- Valeur négative si pas de processus

Création processus

- Exemple
- P1 ----------→P2
- Pid p1=20 et pid p2=30
- i=fork()
- I p2=0 i de p1=30

Accés concurrent aux ressources:

Accès aux mêmes ressources→gestion des droits d'accès.

Solutions :

- **Exclusion mutuelle avec attente active :**

  Tant que (Etat-verrou = = occupé) ;

  Etat-verrou = occupé ;

  Accès à la ressource ;

  Etat-verrou = libre ;

Accés concurrent aux ressources:

- **Désactivation des interruptions :**

  désactiver les interruptions quand un processus veut entrer en section critique et de les réactiver en sortant.

Inconvénient:

Il est généralement dangereux d'autoriser les désactivations d'interruptions car cela pourrait bloquer l'ordinateur.

Accés concurrent aux ressources:

Solution

Pour un processus P0 :

```
While (true)

{

While (A_qui_le_tour != 0) ;

Section_critique ( ) ;
A_qui_le_tour  = 1 ;

Section_non_critique ;

}
```

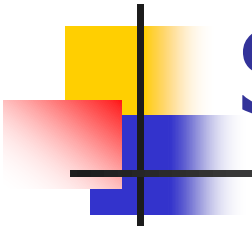
Pour l'autre processus P1 :

```
While (true)

{

While (A_qui_le_tour != 1) ;

Section_critique ( ) ;
A_qui_le_tour  = 0 ;

Section_non_critique ;

}
```

Problèmes classiques de synchronisation

- *Problème des producteurs et des consommateurs :*

```
process producteur;
    begin
    loop
    produire un message;
    tampon.déposer(message);
    end loop;
    end producteur;
process consommateur;
    begin
    loop
    tampon.prélever(message);
    utiliser le message;
    end loop;
    end consommateur;
```



Sémaphore

- inventés en 1965 par Dijkstra
- variable qui contrôle l'accès à une ressource partagée et indique le nombre d'éléments de la ressource qui sont disponibles et maintient une liste des processus bloqués en attente de cette ressource.
- Solutions : créer deux commandes : sleep et wake up

Sémaphore

- **P(S)**
- **{ S.valeur=s.valeur-1;**
- **Si s.valeur<0**
- **Ajouter ce processus à la liste**
- **Sleep() ;**
- **Fin si }**
- **V(S)**
- **{ S.valeur=s.valeur+1;**
- **Si s.valeur<=0 alors Retirer un processus de la liste**
- **Wake up(proc) ;**
- **Fin si }**

❑ Introduction

- Les **threads** appelés aussi « processus légers » sont des mécanismes permettant à un programme de faire plus d'une chose à la fois, ils sont similaires aux processus puisqu'ils représentent tous les deux l'exécution d'un ensemble d'instructions du langage machine d'un processeur .
- Du point de vue de l'utilisateur ,ces exécutions semblent se dérouler en parallèle . Toutefois chaque processus possède ses propres ressources (fichiers, mémoires, etc.),

❑ Création de Threads

- Chaque thread d'un processus est caractérisé par un *identifiant de thread*. Lorsque vous manipulez les identifiants de threads dans des programmes C ou C++, veillez à utiliser le type ***pthread_t***. Exemple : **pthread_t thread_id;**

- Lors de sa création, chaque thread exécute une ***fonction de thread,*** *il* s'agit d'une fonction ordinaire contenant le code que doit exécuter le thread **.**

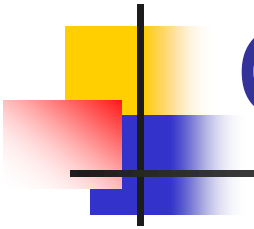
- Lorsque la fonction se termine, le thread se termine également. Sous GNU/Linux, les fonctions de thread ne prennent qu'un seul paramètre de type *void** et ont un type de retour *void**.

✓ **void* print_xs (void* arg) {/* le code que doit exécuter le thread */}**

❑ Création de Threads

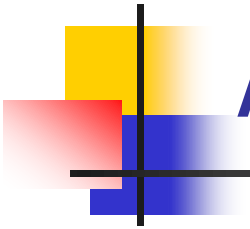
- La fonction ***pthread_create*** crée un nouveau thread. Voici les paramètres dont elle a besoin ,voyons l'exemple suivant:
  - ✓ **pthread_create (&thread_id, NULL, &print_xs, NULL);**

- **L'argument** de la fonction thread est une méthode pratique pour passer des données à un thread. Comme son type est *void**, cependant, vous ne pouvez pas passer beaucoup de données directement en l'utilisant. Une technique couramment utilisée est de définir une structure de données pour chaque argument de thread.



Gestion de mémoire: objectifs

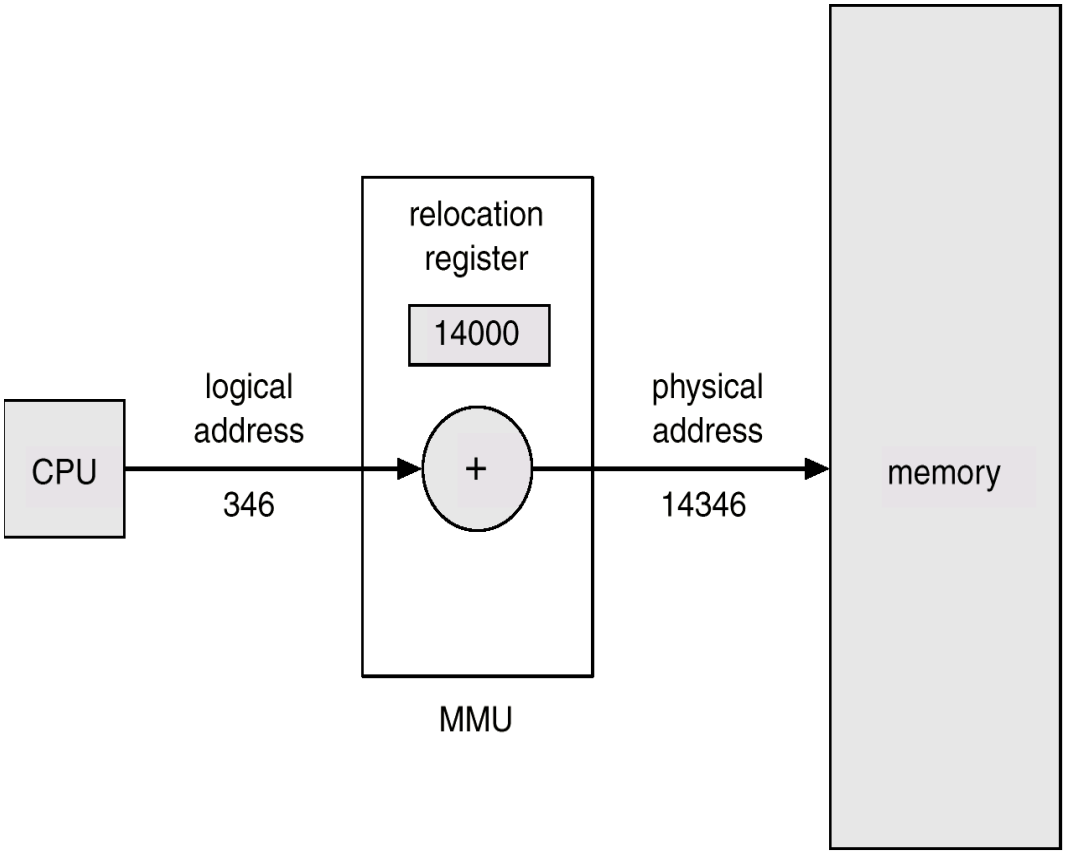
- Optimisation de l ’utilisation de la mémoire principale = RAM
- Les plus grand nombre possible de processus actifs doit y être gardé, de façon à optimiser le fonctionnement du système en multiprogrammation
  - garder le système le plus occupé possible, surtout l’UCT
  - s’adapter aux besoins de mémoire de l ’usager
    - allocation dynamique au besoin



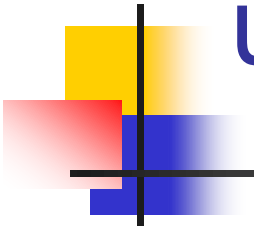
Adresses physiques et logiques

- Mémoire physique: la mémoire RAM de la machine
- Adresses physiques: les adresses de cette mémoire
- Mémoire logique: l'espace d`adressage d'un programme
- Adresses logiques: les adresses dans cet espace
- Il faut séparer ces concepts car normalement, les programmes sont chargés de fois en fois dans positions différentes de mémoire
  - Donc adresse physique ≠ adresse logique

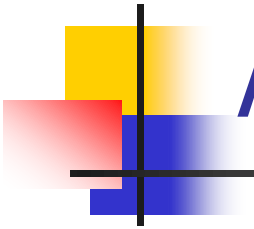
adresses logiques → adr.physiques



Fragmentation: mémoire non utilisée

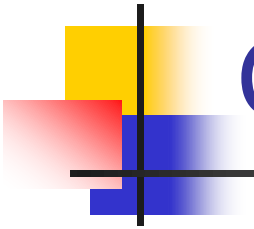


- Un problème majeur dans l`affectation contiguë:
  - Il y a assez d ’espace pour exécuter un programme, mais il est fragmenté de façon non contiguë
    - externe: l`espace inutilisé est entre partitions
    - interne: l ’espace inutilisé est dans les partitions



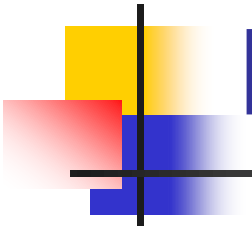
Gestion de la mémoire
Allocation de la mémoire

- Les tables de bits
- Listes chaînées



Gestion de la mémoire virtuelle

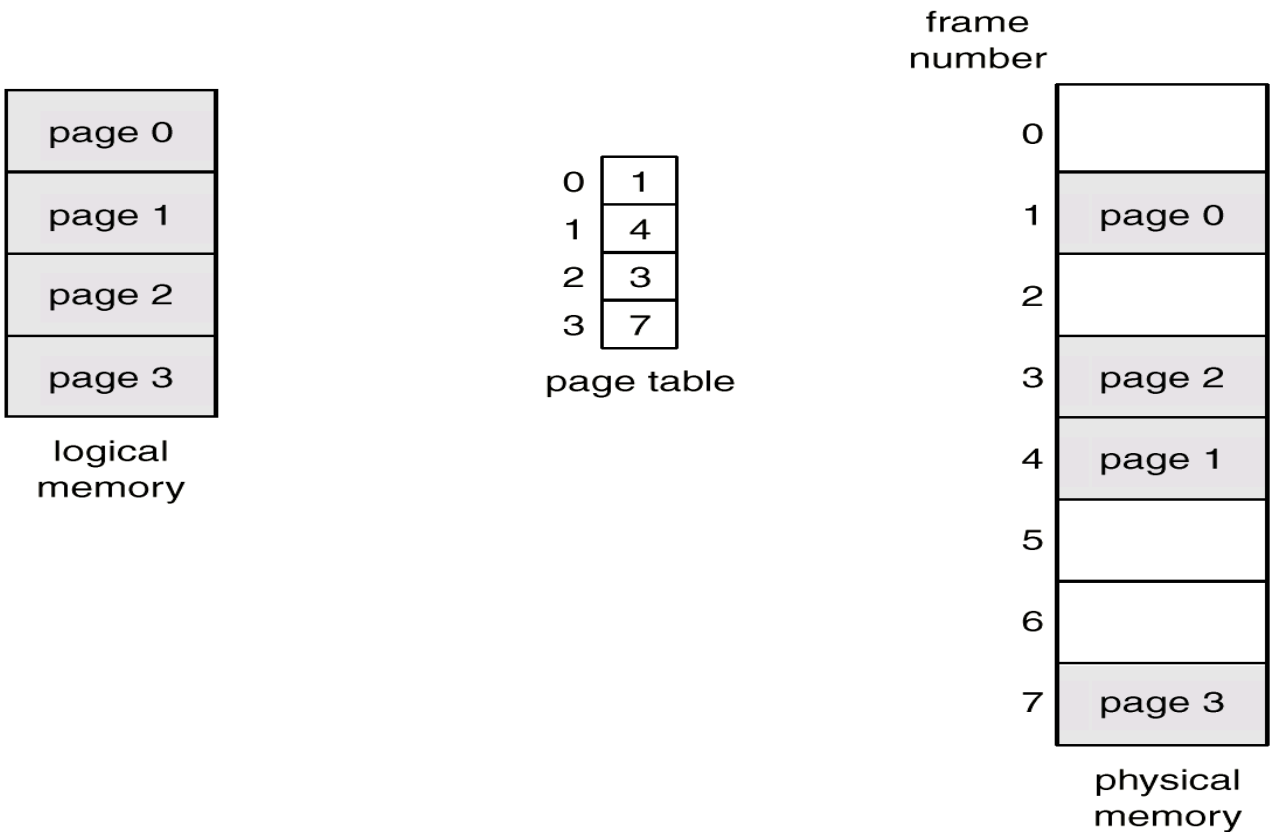
- Principe
- La pagination
- Les algorithmes de remplacement de page



Pagination simple

- La mémoire est partitionnée en petits morceaux de même taille: les pages physiques ou 'cadres' ou 'frames'
- Chaque processus est aussi partitionné en petits morceaux de même taille appelés pages (logiques)
- Les pages logiques d'un processus peuvent donc être assignés aux cadres disponibles n'importe où en mémoire principale
- Conséquences:
  - un processus peut être éparpillé n'importe où dans la mémoire physique.
  - la fragmentation **externe** est éliminée

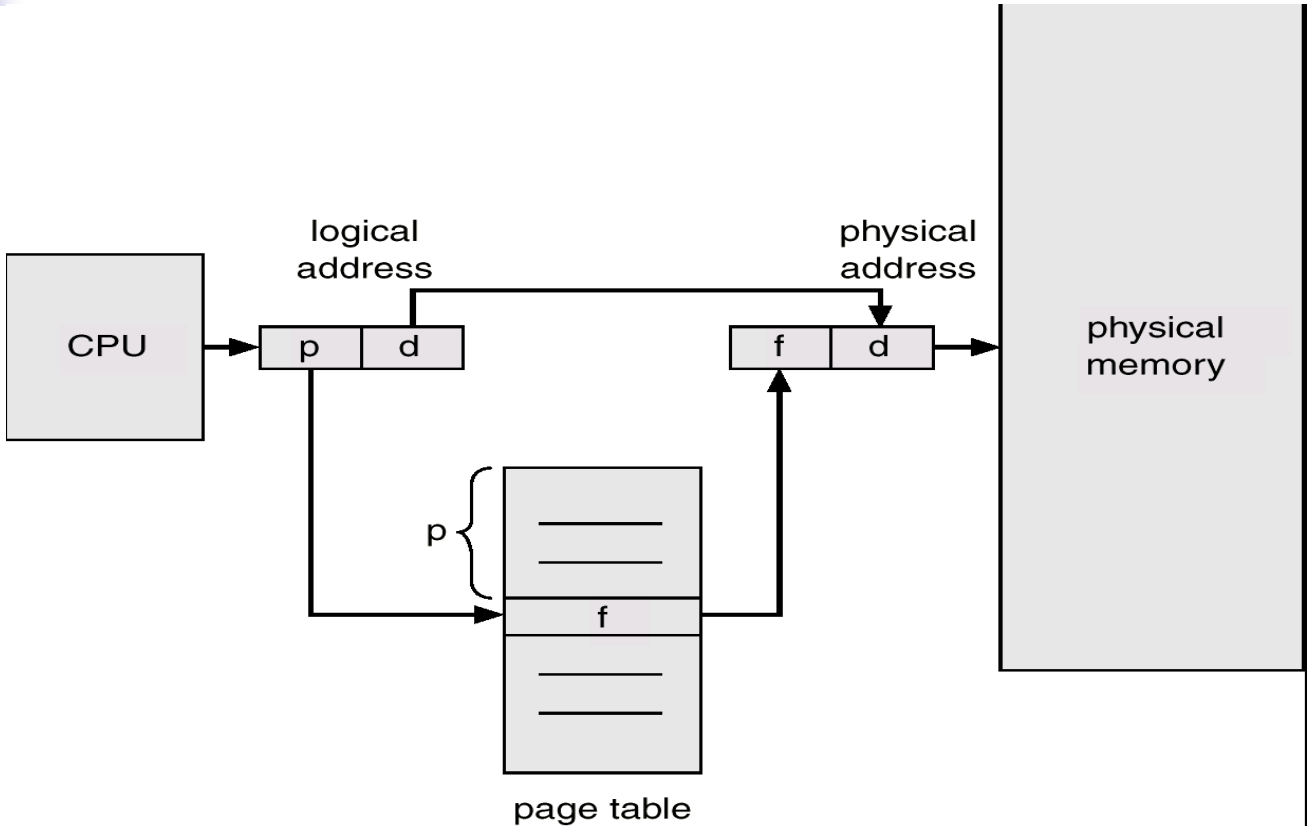
Tableaux de pages



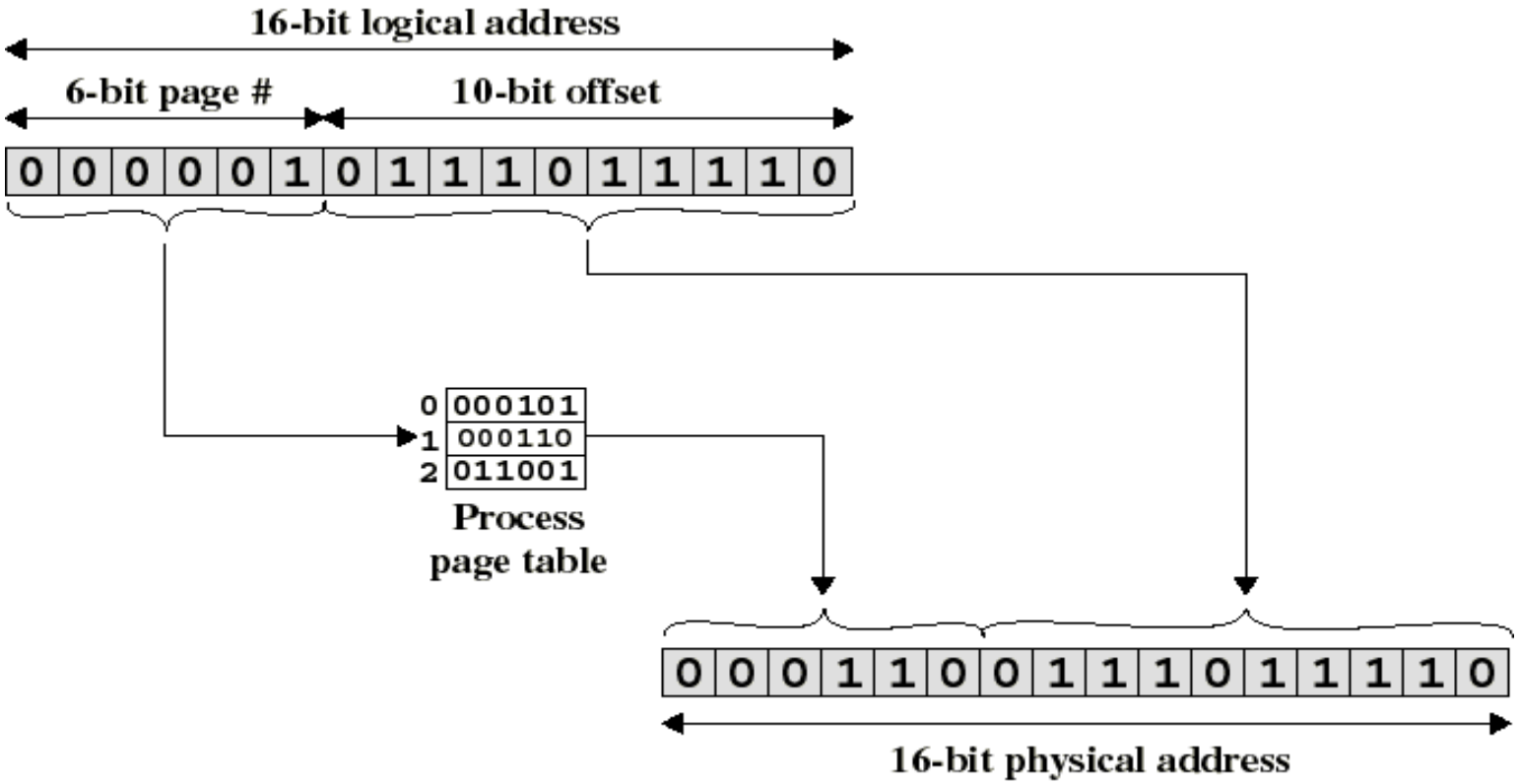
Gestion de la mémoire virtuelle
algorithmes de remplacement de page

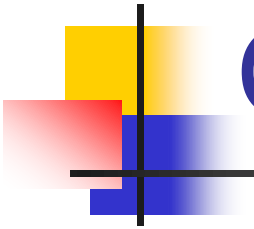
- Page optimale
- Premier entré premier sortie FIFO
- La page la moins récemment utilisé LRU

Mécanisme: matériel



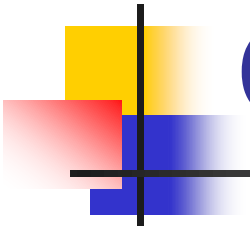
Traduction d'adresse (logique-physique) pour la pagination





Gestion des fichiers

- Allocation des fichiers
- Gestion des blocs libres

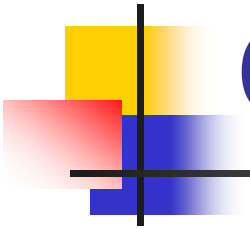


Qu'est ce qu'un fichier

- Collection nommée d'informations apparentées, enregistrée sur un stockage secondaire
  - Nature permanente
- Les données qui se trouvent sur un stockage secondaires doivent être dans un fichier
- Différents types:
  - Données (binaire, numérique, caractères….)
  - Programmes

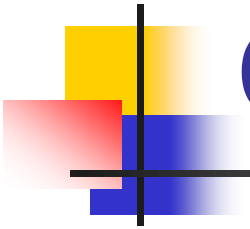
Attributs d'un fichier

- **Constituent les propriétés du fichiers et sont stockés dans un fichier spécial appelé répertoire (directory). Exemples d'attributs:**
  - Nom:
    - pour permet aux personnes d'accéder au fichier
  - Identificateur:
    - Un nombre permettant au SE d'identifier le fichier
  - Type:
    - Ex: binaire, ou texte; lorsque le SE supporte cela
  - Position:
    - Indique le disque et l'adresse du fichier sur disque
  - Taille:
    - En bytes ou en blocs
  - Protection:
    - Détermine qui peut écrire, lire, exécuter…
  - Date:
    - pour la dernière modification, ou dernière utilisation
  - Autres…



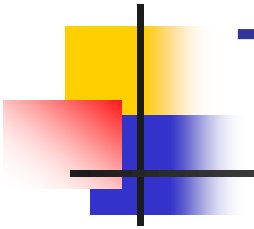
Opérations sur les fichiers

- Création
- Écriture
  - Pointeur d'écriture qui donne la position d'écriture
- Lecture
  - Pointeur de lecture
- Positionnement dans un fichier
- Suppression d'un fichier
  - Libération d'espace
- Troncature: remise de la taille à zéro tout en conservant les attributs



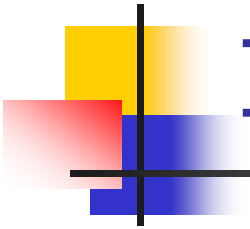
Opérations sur les fichiers

- Ajout d'infos
- Rénommage
- Copie (peut être faite par rénommage)
- Ouverture d'un fichier: le fichier devient associé à un processus qui en garde les attributs, position, etc.
- Fermeture
- Ouverture et fermeture peuvent être explicites (ops *open, close*)
- ou implicites



Types de fichiers

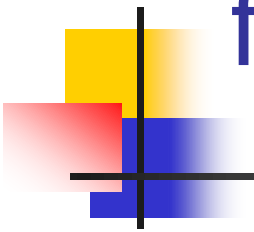
- Certains SE utilisent l'extension du nom du fichier pour identifier le type.
  - MS-DOS: Un fichier exécutable doit avoir l'extension .EXE ou .COM (sinon, le SE refusera de l'exécuter)
- Pour certains SE le type est un attribut
  - MAC-OS: l'application qui crée le fichier définit son type (ex: document Word Perfect)
- Le type n'est pas défini pour certains SE
  - Unix: l'extension est utilisée (et reconnue) seulement par les applications



Informations dans un répertoire

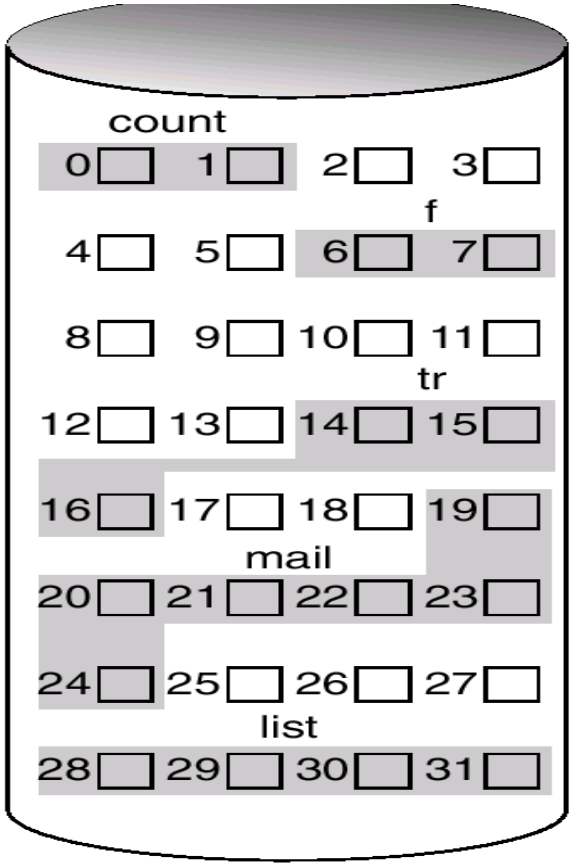
- Nom du fichier
- Type
- Adresse sur disque, sur ruban...
- Longueur courante
- Longueur maximale
- Date de dernier accès
- Date de dernière mise à jour
- Propriétaire
- Protection

Structures de systèmes de fichiers



- Structure de fichiers: deux façons de voir un fichier:
  - unité d'allocation espace
  - collection d 'informations reliées
- Le système de fichiers réside dans la mémoire secondaire: disques, rubans...
- File control block: structure de données contenant de l 'info sur un fichier
- **La mémoire secondaire est subdivisée en blocs de taille fixe et chaque opération d'E /S s'effectue en unités de blocs**

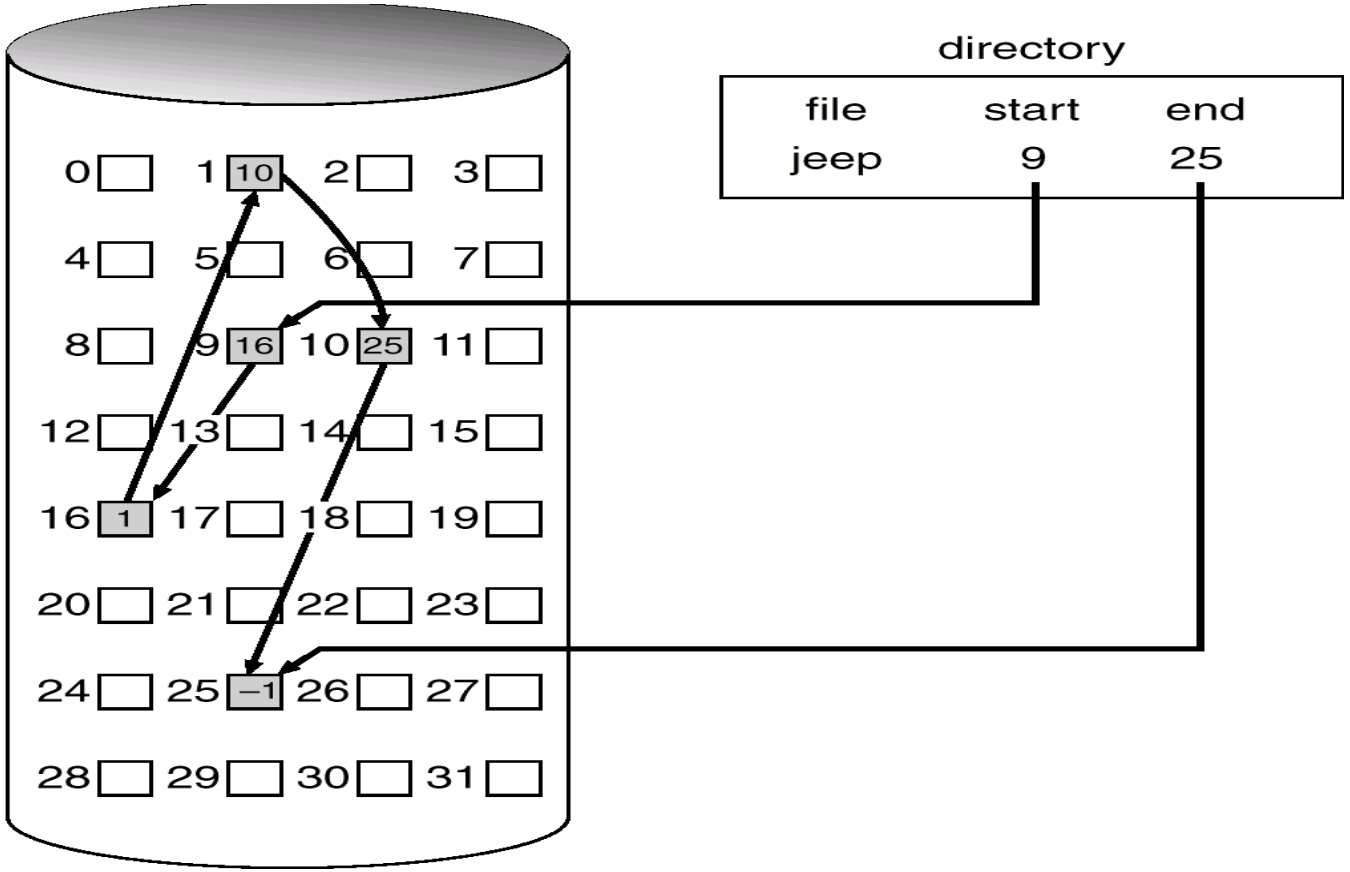
Allocation contiguë sur disque

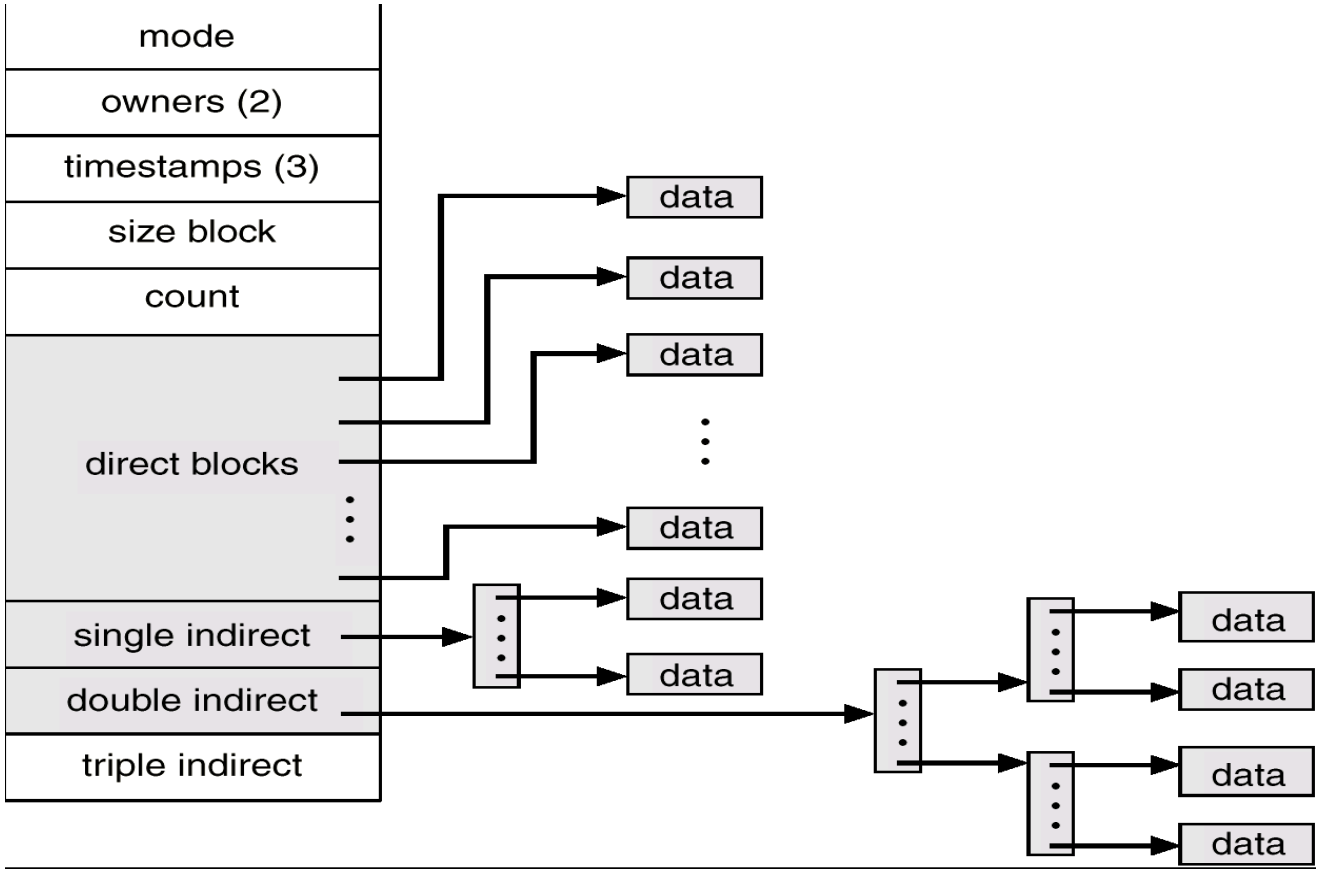


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Allocation enchaînée




mode
owners (2)
timestamps (3)
size block
count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data